QUANTITATIVE EVALUATION OF DESIGN TRADEOFFS IN FILE SYSTEMS

C. P. Wang and V. Y. Lum
Information Sciences Department
IBM Research Laboratory, San Jose, California

ABSTRACT:

The design of a file system has never been a simple nor a straightforward task because of its complexity. Heuristics and experience still play a major role in guiding the design process. To organize the entire design process in a more systematic manner, large scale simulation has proved to be an effective technique. The FOREM models developed during the past several years (specifically for the evaluation of file system designs) represent facilities of this type. This paper utilizes the FOREM model as the principal tool and presents a hypothetical design example dealing with many essential issues of the design process. Evaluation of designs of several other actual file systems are also being carried out and will be reported at a later date. Only through quantitative evaluation can each design decision be arrived at correctly and the possible tradeoffs be identified.

KEY WORDS AND PHRASES

file design, file organization, file simulation and modeling, information retrieval, secondary indexes, data management, secondary keys, inverted file, quantitative design evaluation

I. INTRODUCTION

The complexity of an information system is manifested in the form of at least three major levels of system abstraction each having a set of assorted primitive concepts, design parameters and issues. At the user's problem or logical level, the application analyst deals with sets of objects or entities, their properties and relationships. He may choose a set of definition schemata to declare entities and the pertinent relationships and provide procedures for their manipulation. In this environment, the issues to be dealt with are the user's requirements, his desired output, and the application algorithm. At the access path design level of the systems, the choice of data structure for the data base depends on the combined requirements of all the existing and prospective users of this data base and the projected transaction loads. Here, the data elements may be combined in a logical manner different from that specified by the user's schemata to facilitate search and access. Other facilities such as indexes and cross reference tables are also introduced here to improve the efficiency of finding the appropriate item in the data structure. The data base designer has to provide the proper search algorithms for transforming the users specification of any subset of entities into search steps which are compatible with the data structure of the data base. At the physical level, the chosen structure of the data base must be mapped into the addressing space of the storage device in the host computer and the idiosyncrasies of the data management software have to be taken into account before making the design decisions.

Specific tasks at each level of the design process involve respectively the characterization of the application, the selection of an appropriate file organization, and the determination of many hardware and software parameters to implement that organization. Since the types of applications supported by a file and the resulting transactions are almost unlimited in variety, no single design is equally good for every application. The problem of file design is essentially that of tailoring the file organization to the application in such a manner that some objective function of the system resource requirement, and the aggregated processing time is a minimum. In case the file system provides on-line inquiry capability from remote terminals, additional constraints such as the user's specified response time requirement should also be incorporated in the design.

Many analytical studies have been done in the past on the problem of file design, but most of them concentrated on the filing schemes and the associated access techniques (1,2). More recently, some work has been initiated to treat the file system as a comprehensive problem of design based partially on known optimization techniques (3,4,5). Because of the large number of variables associated with a file system, success is rather limited and is largely confined to files with the simple sequential organization (6,7). Design and optimization of the file system in the general sense seems to remain a distant possibility. At present, the most effective means in carrying out the file system design in a systematic way is through large scale simulation. In this connection, the FOREM (File Organization and Evaluation Model) model has been developed to facilitate the evaluation of various file organizations.

The FOREM model, through the use of many parameters, allows the modeler to specify all three levels of abstraction with sufficient complexity. Extensive use has been made of the model in evaluating many design alternatives at the access path--physical levels, including the organization of data sets, the selection of access methods and many subtle issues such as the blocking factor, the overflow handling technique, the buffering scheme (8), etc. In this paper, we shall present a hypothetical yet quite realistic example of file and application as a vehicle to illustrate the logical sequence of the design process and the associated

issues. The FOREM model provides a convenient probe for the quantitative verification of these issues, their alternatives and possible tradeoffs. It is the authors' intention to accumulate sufficient insight and experience through this method of quantitative evaluation to guide the development of a systematic methodology of file system design and optimization.

II. THE FOREM MODEL

Much of the detailed description on the functional as well as operational aspects of the FOREM model has been discussed elsewhere (9,10). To retain continuity without unnecessary duplication, we shall describe only those features of the model that are particularly relevant for the design issues discussed in this paper.

The FOREM model is a programming facility allowing the modeler to simulate file management systems and the associated dynamic storage and retrieval activities. The facility calculates the total allocated storage space and computes the processing time for each transaction. In its present form it can accommodate more than one hundred types of variables which we shall not attempt to list here. However, it should be beneficial to review at least some essential ones in the framework of the above-mentioned three levels of system abstraction.

The model does provide a rudimentary facility for the specification of user's data elements and their relationships. The elementary unit of data is a field. A contiguous set of fields is called a segment. Different segments can be organized hierarchically to represent complex relationships among data elements. In addition, user's retrieval, update and insertion operations can also be expressed in tabular form with qualification on keys as well as on other field values. In a qualification expression, the allowable predicates include equal, greater than, less than and range. The specification at users level is essentially independent of the access path or physical levels of the file. For the specification of the data base, the notion of access-path-level data sets is introduced. All segments and their relationships specified in the user's schemata can be mapped into a single or several data sets at the discretion of the designer. Facilities are also available for the specification of indexes and cross reference tables which are not part of the user's schemata. Basic mechanisms for the construction of search strategies in tabular form are also provided to transform the user's storage and retrieval specifications into search steps directed to the appropriate data sets.

To enable the users to design the storage implementation of the access-path-level data sets, the model is capable of simulating a number of storage devices such as tape, drum, data cell and disk, and several standard access techniques supported by the IBM 360 operating system. These include sequential, indexed sequential and direct access methods. Other storage and access technique related parameters such as blocking factor, overflow handling technique, buffering mode, etc. are also provided as design variables.

As outputs, the model calculates the storage space occupied by each data set and the processing time (elapsed time) required for each user transaction. The designer can then use these two outputs as basic indicators to guide his entire design process.

III. APPLICATION DESCRIPTION

The best approach to provide an illuminating insight to the entire design process and the attainable tradeoffs is to follow through the design steps of a file system accompanied by the rationale behind each step. The first step to start that process is to provide a parameterized description of the application and the associated collection of data items. Consider a hypothetical application dealing with three sets of objects called VENDORS, PARTS, and PROJECTS. Each vendor is characterized by a unique identification NUMBER, a NAME, and an ADDRESS. Each part has the attributes PART NUMBER and DESCRIPTION while each project has the attributes PROJECT NUMBER and PROJECT NAME. In addition, the application is also dealing with the 3-tuple relationship (VENDOR NAME, PART NUMBER, COLOR) and the 4-tuple relationship (VENDOR NUMBER, PART NUMBER, PROJECT NUMBER, QUANTITY). The user's declaration can conceivably be given as

FIELD DECLARATION

LEVEL	FIELD NAME	ENCODED CHARACTER LENGTH
0	VENDOR NUMBER (KEY)	6
0	NAME	10
0	ADDRESS	60
1	PART NUMBER	8
1	COLOR	8
1	DESCRIPTION	20
2	PROJECT NUMBER	8
2	QUANTITY	4
2	PROJECT NAME	16

SEGMENT DECLARATION

SEGMENT NAME	FIELDS IN SEGMENT	SUPERIOR SEGMENT NAME	AV. NO. OF REPETITIONS
S0	VENDOR NUMBER	--	1
	NAME		
	ADDRESS		
S1	PART NUMBER	S0	2
	COLOR		
	DESCRIPTION		
S2	PROJECT NUMBER	S1	3
	QUANTITY		
	PROJECT NAME		

One occurrence of such a record as viewed by the user is shown in Fig. 1. It is interesting to note here that there exist many-to-many relationships among the entities called VENDOR, PART, and PROJECT and the above specification is certainly not the only one possible. The question of what

are the most appropriate schemata for a given application is an interesting area of investigation beyond the scope of this paper. Here, we simply postulate that the particular choice of such a declaration is a result of a careful analysis of the application. To preserve the simplicity of the problem, let us further assume that it is the only application supported by the file management system.

In addition to the segment declaration given above, information about the field values, the inquiry and maintenance activities, and the corresponding occurrence frequencies are also necessary to provide a complete characterization of the file and the associated applications. For those fields used in the selection criteria of user's storage and retrieval requests, their value sets and occurrences of each value in the file enable a more accurate determination of the total I/O activities to be initiated in connection with each user's transaction and, hence, a more accurate estimate of the processing time of that transaction.

The dynamic inquiry and update activities generated by the user's application of this file are represented by the set of hypothetical queries given in Table I. Although it appears that the query set is made up in an arbitrary manner, it can be divided into four broad classes, namely (i) qualification on one or more fields in the master segment and output either the entire record or segments at any subordinate level, (ii) joint qualification on fields in both the master segment and the subordinate segment and output either the entire record or any segment, (iii) qualification on fields in one or more levels of subordinate segments and output the entire record or the master segment, and (iv) given record key update the entire record or some fields in the record. It should be noted that the frequency of occurrence associated with each inquiry and update operation has the significance of weighting function in their effect on the outcome of the design. No record insertion operation is given here. It is assumed that the file is periodically reorganized in a batch maintenance mode when sufficient number of new records are accumulated for insertion.

It can be observed from Table I that only four fields, "VENDOR NAME," "PART NO.," "COLOR" and "PROJECT NO." are used in the selection criteria of any user's transaction. To generate the number of records retrieved from the qualification criterion of a query, the following two sets of hypothetical field value distributions are provided in Table II. Although these two sets of distributions differ only in the field "COLOR," it should be interesting to observe their effect on the result of the design.

The application description is a crucial aspect of the design process. Great care should be exercised to ensure that the description is completely independent of the access path and physical implementations of the files. The types of characteristics given above provide a framework for extracting appropriate parameters.

IV. EXAMPLES OF DESIGN

The first major design issue to be resolved is the selection of a logical organization for the data base. In the general case, a data base may support a number of applications and conceivably has an organization quite different from that specified by the user's schemata. The problem of selecting the best schemata for the data base taking into account the requirements and activities of all supported applications is an extremely difficult and unresolved problem. It is certainly the key to a systematic approach of data base system design and optimization. In the hypothetical example considered in this paper, the situation is much simpler because there is only one application associated with the data base. We shall explore two access-path-level organizations in more detail in the following:

A. Choice of Data Base Organization

The first access-path-level organization of the data base to be investigated has the same format as that declared by the user. Assuming that each encoded character of every field takes one byte (8 bits) of space, the length of the record can then be readily calculated. We shall also postulate that there are 10^5 records in the data base. This set of records can be sorted in collating sequence of the keys into an access-path-level data set. It should be noted that the sorting order introduced in the data base may be quite different from that envisioned by the user.

Next, let us consider the data set so specified and the search technique required in finding records in the file which satisfy the selection criterion in any given query. Of course, the set of records satisfying a given query listed in Table I can be found by scanning the entire file sequentially in a record by record manner. But for fast retrieval, via key, an index table of record keys can be built. It is generally recognized that cross reference indexes organized into separate data sets can also be created on those fields which are used in the qualification criterion of any query. From the query Table I, it is clear that four fields "VENDOR NAME," "PART NO.," "COLOR," and "PROJECT NO." are potential candidates for the creation of cross reference indexes. However, extra maintenance effort is required to add a value or a key entry or both to each of the cross reference indexes during the insertion of a new record to the file. When updating the field value of a record, extra work in modifying the cross reference table may be necessary if that field is indexed. In any event, the merging time of two long key lists obtained from two different cross reference indexes for joint qualification on two or more fields may be quite substantial. Therefore, the design problem is to balance the benefits of reduced retrieval times against the drawbacks of extra index processing and maintenance efforts.

Consider now the processing time for the query Q10 in Table I. Suppose all four fields mentioned above are indexed. Then there are three

possible search strategies to find all the qualifying records. The first strategy starts from the cross reference indexes on both fields "PART NO." and "COLOR", retrives one qualifying record key list from each and then performs the intersection operation. The end result is a shorter key list which can subsequently be used for record retrieval. The second strategy retrieves only the record key list from the index on the field "PART NO." and then uses that for the selection of record. The third search strategy, of course, uses only the cross reference index on the field "COLOR." Since the first strategy in general produces the shortest key list for record retrieval, it is usually but not always the best strategy in most situations. However, for the hypothetical file under consideration, the field "COLOR" has only 4 distinct values. The length of the key list associated with any value entry in the cross reference index on "COLOR" is at least 10^4 key entries or more. Consequently, a great deal of time may be consumed in key list accessing and merging. All three cases of cross reference indexing will be evaluated quantitatively.

B. Selection of Physical Structure

The next stage of design deals primarily with the physical structures and their implementations. The choice is limited, of course, to the available options of the software and hardware facilities provided by the host computer. In this example, we shall consider only the disk storage device because of the real time nature of the inquiries. The design evaluation tool, the FOREM model, is able to simulate operations of the three principal access methods, namely the sequential, the indexed sequential and the direct access methods. Qualitatively speaking, the sequential organization is best suited for batch-type sequential processing. It is not suited for inquiries with qualification on any field other than the key field. The direct organization gives rise to the shortest retrieve time under the random retrieval mode, but yields relatively poor performance under sequential processing. In other words, for any retrieval criterion based on a non-indexed field, a time consuming bucket-by-bucket search will result. When there is substantial insertion of new records into the file, the direct organization is also likely to deteriorate in performance due to the necessity of chaining the overflow records from the home bucket. The indexed sequential organization provides the capability for both sequential and random processing and can tolerate moderate amounts of insertion without severe performance degradation. However, the data set does have to be reorganized periodically when the overflow area becomes heavily populated. For the set of queries given in Table I, the index sequential organization seems to be the appropriate choice.

Besides the selection of an organization for a data set, there are several additional design parameters which must be determined. For the disk storage device, records must be blocked; the blocking factor determines the number of interblock gaps on a track and hence determines the number of records that can be stored in a track. It also determines the size of buffers that must be allocated in core. The average record length in the ISAM data set is 316 bytes and is grouped 5 records to a block on a 2314 disk storage device. Since there are 7294 bytes per track on the 2314 disk, the choice will produce four blocks to a track. The cylinder overflow mode is assumed here and there are 5% of the records in the overflow area. In addition, the four cross reference indexes are also organized into index sequential data sets with record lengths from 8 to 20 bytes, 100 records per block and 1% overflow.

After all the design parameters were selected and entered into the FOREM evaluation model, the total retrieval time for the query Q10 was computed for the three different search strategies discussed earlier. The resulting retrieval times were 52.7 sec, 21.4 sec, and 6896 sec, respectively, favoring the strategy using only indices on "VENDOR NAME," "PART NUMBER," and "PROJECT NUMBER."

It might be argued that the principal reason for the creation of an index on the field "COLOR" is to eliminate a time consuming sequential search of the complete file, when "COLOR" is the sole field used in the selection criterion of an inquiry. An inspection of the query set in Table I reveals that the appearance of "COLOR" in the selection criterion of any query is always accompanied by another field. This observation essentially leads to a conjecture that the creation of an index on "COLOR" does not serve any useful purpose. This conjecture has been borne out by the results obtained from many simulation runs. Table III gives the total time for processing the inquiries and updates in the query set for three different cases of index selections: (i) all four fields, "VENDOR NAME," "PART NO.," "COLOR," and "PROJECT NO." indexed, (ii) all except "PART NO." are indexed, and (iii) all except "COLOR" are indexed. Clearly the best performance results when no index is created on "COLOR." This simple example has demonstrated that although the creation of an index on a field is generally viewed as beneficial, the benefit cannot be ascertained without quantitative evaluation.

Now the dependence of the system performance on the content of the field can be readily demonstrated. Of course, the distribution of a field in the real world can never be arbitrarily assigned. It is an intrinsic characteristic of the file content. To demonstrate this dependence, let us consider an alternate field value distribution given in Table II. All other conditions remain the same as those in the previous design. For this alternate distribution, the total processing times for all the inquiries and updates are recomputed using the FOREM model and placed in Table IV. The results indicate that total service time of the entire query set has been reduced and that, contrary to the earlier conclusion, an implementation with indexes on all four fields now gives the best performance.

This example clearly illustrates our earlier contention that the content distribution character-

ization should be a part of the application description.

C. An Alternate Data Base Organization

It would be beneficial to examine another choice of the logical organization of the data base which does not have the identical notion of records as that envisioned by the user. For example, each hierarchical record envisioned by the user (Fig. 1) can be mapped into two different types of records of the data base with one VENDOR segment and two PART segments grouped together as a record of one type and each PROJECT segment as a record of another type. Figure 2 depicts the mapping and the resulting record types. A new key or identification number must be assigned to records of the second type. Then the records of each type are grouped together in sorted collating sequence of the keys or identification number to form two data sets. The first data set has 10^5 records and a record length of 148 bytes. The second data set has 6×10^5 records of 28 bytes per record. It may have a sorted sequence different from that shown in Fig. 2. However, to preserve linkage among segments belonging to the same original user record, pointers are imbedded in every record of the first data set to link to the appropriate records in the second data set. As far as the choice of the cross reference indexes is concerned, creation of indexes on field "VENDOR NAME," "PART NO.," "COLOR," and "PROJECT NO." seems still a reasonable design decision.

The remaining task for the designer is, of course, to choose a physical organization to place the two data sets on a disk storage device such as 2314. Here we examine only two cases. In the first case, both data sets are organized as indexed sequential data sets. The first data set has a blocking factor of 8, while the second data set has a blocking factor of 60. Both data sets are assumed to have 5% overflow records. In the second case, the first data set is organized into an indexed sequential data set while the second data set is organized as a direct access data set.

Let us now consider the case that both data sets are indexed sequential. For such an organization of the data base, the query processing time depends critically on the query type and levels of segments to be retrieved. For example, to retrieve a complete user record, the transaction must first go to the master segment of the user record in the first ISAM data set, and then follows the pointers to retrieve all the subordinate segments in the last level which is also an ISAM data set. On the average, there will be six accesses to the last level data set per user record retrieval. Table V presents the processing times for individual query as well as for the entire query set for this implementation. The results show that overall processing time will be increased substantially.

Next let us consider the case that the second data set is organized as direct access data set. All the processing times for the query set were recomputed as given in Table V. The result clearly favors the direct access organization for the second data set.

A comparison of Table V with Table III shows that for some queries Q2, Q3, Q5, Q7, Q12, Q15, and Q16, the processing times are shorter in the alternate data base organization. But for the entire set of queries, the first data base organization gives rise to shorter aggregate processing time. This may not always be the case. When both qualifications and output are confined to segments in the first two levels, the two data set organizations may become favorable. Further, there are several other organizations not considered in this paper. For example, it may be possible that last level segments belonging to the same superior segment are stored near each other and cause a reduction in disk arm movement during their retrievals. This kind of refinements, which may well reverse our conclusions, require much more detailed analysis than that given in this paper.

V. CONCLUSIONS

The technique of evaluating alternate designs of file organizations based on the given file and application characteristics has been demonstrated through the example considered in this paper. Conventional hand calculations and imprecise rules of thumb become inadequate in dealing with complex problems of this magnitude. The usefulness of a quantitative evaluation tool such as the FOREM model can never be overemphasized.

The design alternatives, of course, represent only a small fraction of a wide variety of issues and tradeoffs encountered in designing any large file system. However, two observations should be noted here. First, in a complex design problem involving a large number of variables, it is not unusual that at each level of decision the resulting performance depends critically on the disposition of a small subset of variables but is relatively insensitive to the choice of the other variables over a wide range of values. Identification of the sensitivities of variables and their corresponding ranges of values would certainly be of positive value to the design process. Secondly, since the number of variables is large, an exhaustive enumeration of all possible combinations is prohibitively expensive even if each variable is assigned only a few trial values. It is thus imperative to devise some heuristic algorithms to reduce the number of combinations to be tested. The quantitative evaluation methodology presented in this paper represents an initial step in those directions.

ACKNOWLEDGMENT

The authors would like to thank M. Senko for many helpful discussions on the stratification of system abstractions and his suggestion of the term "the access path design level." The work is partially supported by the Rome Air Development Center through contract AF 30602-70-C-0114.

REFERENCES

(1) BUCHOLZ, W. File organization and addressing. IBM Systems Journal, 2 (June 1963) p. 86-111.

(2) GHOSH, S.P.; SENKO, M.E. File organization: on the selection of random access index points for sequential files. Journal of the ACM, 16:4 (October 1969) p. 569-579.

(3) TEICHROEW, D. A review of system design techniques. ISDOS Working Paper 17, University of Michigan, January 1969.

(4) McCUSKEY, W.A. Toward the automatic design of data organization for large scale information processing systems. Ph.D. thesis, Case Western Reserve University, January 1969.

(5) NUNAMAKER, JR., J.F. On the design and optimization of information processing systems. Ph.D. thesis, Case Western Reserve University, June 1961.

(6) LANGEFORS, B. Some approaches to the theory of information systems. BIT, 3 (1963) p. 229-254.

(7) DAY, R.H. On optimal extracting from a multiple file data storage system: An application of integer programming. Journal Op. Res., 13:3 (May-June 1965) p. 482-94.

(8) SENKO, M.E.; LING, H.; LUM, V.Y.; MEADOW, H.R.; BRYMAN, M.R.; DRAKE, R.J.; MEYER, B.C. File design handbook. Contract Report AF 30602-69-C-0100, November 1969.

(9) SENKO, M.E.; ABRAHAM, C.T.; GHOSH, S.P.; LUM, V.Y.; OWENS, P.W.; POMPER, I.H.; BAKER, F.T.; SCHENKEN, J.D.; WALKER, T.P. Formatted file organization techniques. Final Report AF 30602-4088, May 1967.

(10) SENKO, M.E.; LUM, V.Y.; OWENS, P.J. A file organization evaluation model - FOREM. 1968 IFIP Congress Proceedings, p. C19-23.

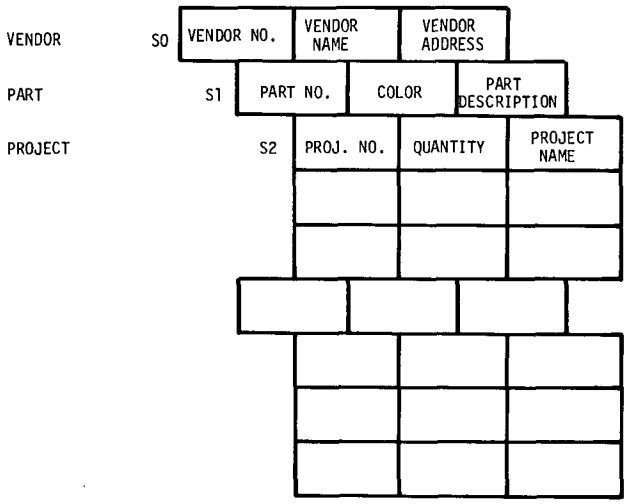
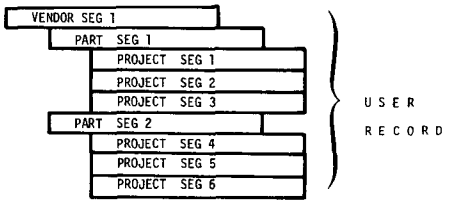


Figure 1. Schematic Representation of a User Record



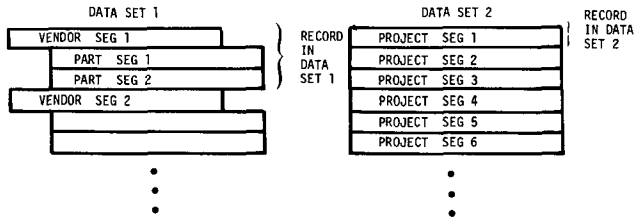


Figure 2. Assignment of Segments of a User Record to Data Set Records

Query No.	Description	Frequency
Q1	Given VENDOR NUMBER=1234, Retrieve Record*	9
Q2	VENDOR NAME=SMITH, PART NO.=01234, Retrieve Record	1
Q3	VENDOR NAME=SMITH, PART NO.=00012, LIST COLOR	1
Q4	VENDOR NAME=JONES, COLOR=RED, Retrieve Record	1
Q5	VENDOR NAME=APPLE, COLOR=BLUE, LIST VENDOR NO. & PART NO.	1
Q6	VENDOR NAME=APPLE, PROJECT NO.=2135, Retrieve Record	1
Q7	VENDOR NAME=SMITH, PROJECT NO.=4275, LIST VENDOR NO. and PARTS SUPPLIED	1
Q8	VENDOR NAME=JONES, Retrieve Record	1
Q9	COLOR=RED, PROJECT NO.=2135, LIST VENDOR NO. & PART NO.	1
Q10	PART NO.=00012, COLOR=BLUE, Retrieve Record	1
Q11	PART NO.=00012, Retrieve Record	1
Q12	PROJECT NO.=3124, LIST VENDOR NAMES SUPPLYING TO THAT PROJECT	1
Q13	Given VENDOR NO.=6337, Update COLOR	2
Q14	Given VENDOR NO.=2184, Update VENDOR ADDRESS & COLOR	2
Q15	Given VENDOR NO.=3526, LIST VENDOR NAME & COLOR	1
Q16	Given VENDOR NO.=7283, LIST PARTS SUPPLIED	1
Q17	Given VENDOR NO.=1124, Update Record	2

* RECORD here refers to the record format in Figure 1.

Table I

Query Number	Frequency	Processing Time In Seconds: Vendor Name Part No. Indexed Color Project No.	Vendor Name Indexed Color Project No.	Vendor Name Part No. Indexed Project No.
Q1	9	2.43	2.43	2.43
Q2	1	3.5	110	3.5
Q3	1	1.54	110	1.54
Q4	1	67.8	67.8	22.8
Q5	1	45.8	45.8	8.87
Q6	1	1.10	1.10	1.10
Q7	1	6.03	6.03	6.03
Q8	1	22.8	22.8	22.8
Q9	1	93.6	93.6	60.0
Q10	1	52.7	6896	21.4
Q11	1	21.4	686	21.4
Q12	1	39.6	39.6	39.6
Q13	2	3.14	1.86	1.24
Q14	2	3.92	2.60	2.60
Q15	1	0.27	0.27	0.27
Q16	1	0.27	0.27	0.27
Q17	2	7.81	6.50	6.50
Total Time:		373.17	8092.09	221.76

Table III

FIELD VALUE ITEM	FREQUENCY OF OCCURRENCE: CASE I Vendor Name	Part No.	Color	Proj. No.	CASE II Vendor Name	Part No.	Color	Proj. No.
V1	54	70	43,210	115	54	70	10,000	115
V2	150	210	31,566	70	150	210	10,000	70
V3	125	520	12,370	45	125	520	10,000	45
V4	80	305	12,854	375	80	305	70,000	375
V5	99,591	78	--	65	99,591	78	--	65
V6	--	1,003	--	913	--	1,003	--	913
V7	-	724	--	4,890	--	724	--	4,890
V8	--	97,090	--	9,672	--	97,090	--	9,672
V9	--	--	--	10,027	--	--	--	10,027
V10	--	--	--	12,300	--	--	--	12,300
V11	--	--	--	25,798	--	--	--	25,798
V12	--	--	--	35,642	--	--	--	35,642

In Table II the total occurrences in the Part No., Color, and Project No. fields have been normalized to 10^5. A multiplication factor of 2 should be used for Part No. and Color and a multiplication factor of 6 should be used for the Proj. No.

Table II

Query Number	Frequency	Processing Time In Seconds: Vendor Name Part No. Indexed Color Project No.	Vendor Name Indexed Color Project No.	Vendor Name Part No. Indexed Project No.
Q1	9	2.43	2.43	2.43
Q2	1	3.50	110	3.50
Q3	1	1.54	110	1.54
Q4	1	67.8	19.8	22.8
Q5	1	45.8	1c.9	8.82
Q6	1	1.10	1.10	1.10
Q7	1	6.03	6.03	6.03
Q8	1	22.8	22.8	22.8
Q9	1	93.6	27.6	60.0
Q10	1	52.7	2469	21.4
Q11	1	21.4	686	21.4
Q12	1	39.6	39.6	39.6
Q13	2	3.14	1.86	1.24
Q14	2	3.92	2.60	2.60
Q15	1	0.27	0.27	0.27
Q16	1	0.27	0.27	0.27
Q17	2	7.81	6.50	6.50
Total Time:		197.07	3522.2	221.76

Table IV

Query Number	Frequency	Processing Time In Seconds		
		One Data Set	Two Data Sets Both ISAM	Two Data Sets One ISAM, One DAM
Q1	9	2.43	11.2	4.17
Q2	1	3.50	12.4	3.24
Q3	1	1.54	1.49	1.49
Q4	1	19.8	179	99.5
Q5	1	16.9	44.6	44.6
Q6	1	1.10	1.10	1.10
Q7	1	6.03	3.52	3.52
Q8	1	22.8	181	63.66
Q9	1	27.6	74	58.2
Q10	1	19.5	93.2	62.1
Q11	1	21.4	92.4	37.5
Q12	1	39.6	34.5	14.7
Q13	2	3.14	3.14	3.14
Q14	2	3.92	3.92	3.92
Q15	1	0.27	0.18	0.18
Q16	1	0.27	0.18	0.18
Q17	2	7.81	10.6	8.51
Total Time:		373.17	730.63	409.71

Table V